(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平６－５４１７６

(43)公開日　平成６年（１９９４）２月２５日

(51)Int.Cl.*	識別記号	庁内整理番号	ＦＩ	技術表示箇所
H04N 1/40		D 9068-5C		
G06F 15/62	310	A 8125-5L		
15/66	310	8420-5L		
15/68	310	9191-5L		
H04N 1/46		9068-5C		

審査請求　未請求　請求項の数４　（全２４頁）　最終頁に続く

(21)出願番号　特願平４－２０５２４６

(22)出願日　平成４年（１９９２）７月３１日

(71)出願人　０００００１００７
キヤノン株式会社
東京都大田区下丸子３丁目３０番２号

(72)発明者　横澤　良和
東京都大田区下丸子３丁目３０番２号　キヤノン株式会社内

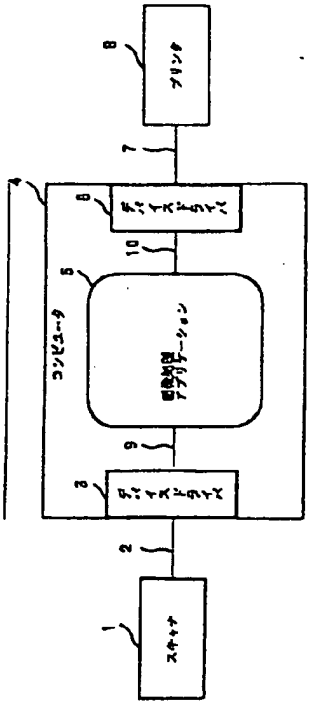
(74)代理人　弁理士　大塚　康徳　（外１名）

(54)【発明の名称】色処理方法

(57)【要約】

【目的】　異なる色空間を有する複数のデバイスを１つの仮想デバイスとして利用する事ができる色処理方法を提供する。

【構成】　スキャナ１からコンピュータ４に対してどの様な色空間を採用しているか、かつ色空間を変換する機能があるかどうかを問い合わせ、コンピュータ４が色空間を変換する機能がある場合には、カラー画像データをコンピュータ４の色空間に変換するパラメータと共に伝送し、コンピュータ４からプリンタ８に対してどの様な色空間を採用しているか、かつ色空間を変換する機能があるかどうかを問い合わせ、プリンタ８が色空間を変換する機能がある場合には、カラー画像データを、プリンタ８の色空間に変換するパラメータとスキャナ１から受信したパラメータとを演算する事によって得られるパラメータと共に伝送する。



1

【特許請求の範囲】

【請求項１】　画像の入出力デバイス毎にデバイスドライバと、各デバイスドライバにそれぞれ対応した入出力デバイスの色補正手段とを有し、カラー画像データを扱うコンピュータシステムの色処理方法であって、

コンピュータ上のアプリケーションから各デバイスドライバに指令して入出力デバイスの色補正を行い、

その色補正されたデータを各デバイスドライバ自身が保管する事を特徴とする色処理方法。

【請求項２】　第１のデバイスから、第２のデバイスにカラー画像データを伝送するシステムの色処理方法であって、

第１のデバイスから第２のデバイスに対してどの様な色空間を採用しているか、かつ色空間を変換する機能があるかどうかを問い合わせ、

第２のデバイスが色空間を変換する機能がある場合には、カラー画像データを第２のデバイスの色空間に変換するパラメータと共に伝送し、

第２のデバイスが色空間を変換する機能がない場合には、カラー画像データを第２のデバイスの色空間に変換して伝送する事を特徴とする色処理方法。

【請求項３】　第１のデバイスから、第２のデバイスにカラー画像データを伝送するシステムの色処理方法であって、

第１のデバイスから第２のデバイスに対してどの様な色空間を採用しているか、かつ色空間を変換する機能があるかどうかを問い合わせ、

第２のデバイスが色空間を変換する機能がある場合には、カラー画像データを両デバイス間の標準色空間に変換するパラメータと共に伝送し、

第２のデバイスでは標準色空間から自己の固有の色空間に変換するパラメータと前記第１のデバイスから受信したパラメータとを演算する事によって得られるパラメータとから、前記第１のデバイスから受信したカラー画像データを第２のデバイスに固有の色空間に変換し、

第２のデバイスが色空間を変換する機能がない場合には、カラー画像データを第２のデバイスの色空間に変換して伝送する事を特徴とする色処理方法。

【請求項４】　第１のデバイスより入力されたカラー画像データの色空間を補正して第２のデバイスへ出力するシステムの色処理方法であって、

第１のデバイスよりカラー画像データを入力し、

前記第１のデバイスに固有の色空間を標準色空間に変換する第１のパラメータと標準色空間を第２のデバイスに固有の色空間に変換する第２のパラメータとを演算し、

該演算結果に基づいて前記カラー画像データの色空間を補正し、

該補正された色空間のカラー画像データを前記第２のデバイスへ出力する事を特徴とする色処理方法。

【発明の詳細な説明】

2

【０００１】

【産業上の利用分野】本発明は、例えばコンピュータからプリンタ等の出力デバイスにカラー画像を印刷する時、スキャナや電子カメラ等の入力デバイスからコンピュータにカラー画像を取り込む時、及びコンピュータから他のコンピュータにカラー画像を伝送する時に各デバイス固有の色空間特性の違いを補正する色処理方法に関するものである。

【０００２】

【従来の技術】例えば、コンピュータ，スキャナ，電子カメラ及びプリンタ等の各種デバイスにおいて同じＲＧＢの色成分を表現できても、各色成分毎の感度は各デバイス毎に微妙に違っており、ユーザの意図したプリント結果が得られない場合が多い。また、例えば１台のコンピュータが複数のプリンタを切り換えて使う場合には、プリンタ毎の色再現が異なる場合が多く、印刷結果がまったく同じにはならない場合が多い。

【０００３】この様な場合、通常、演算処理能力の高いコンピュータにおいて、各デバイス毎に適切な色補正を行っている。この時、アプリケーションソフトの対応方法としては、何らかのメニューを表示させ、そのメニューから必要なプリンタ或いはスキャナ等の固有の色空間を選択させる方法が一般的に行われている。

【０００４】

【発明が解決しようとする課題】しかしながら、上記従来例では、本質的に以下のような欠点があった。

（１）プリンタやスキャナの種類が増えてくると、アプリケーションソフト側での対応が取れなくなる。

（２）色補正が、アプリケーションを操作する者の責任になる。つまり、ユーザが色補正しなければならない。

【０００５】上述の問題は、特にネットワークを介して複数のスキャナやプリンタを複数のコンピュータで共同利用する際に深刻な問題であった。本発明は、上記課題を解決するために成されたもので、異なる色空間を有する複数のデバイスを１つの仮想デバイスとして利用する事ができる色処理方法を提供することを目的とする。

【０００６】

【課題を解決するための手段】及び

【作用】上記目的を達成するために、本発明の構成は、画像の入出力デバイス毎にデバイスドライバと、各デバイスドライバにそれぞれ対応した入出力デバイスの色補正手段とを有し、カラー画像データを扱うコンピュータシステムの色処理方法であって、コンピュータ上のアプリケーションから各デバイスドライバに指令して入出力デバイスの色補正を行い、その色補正されたデータを各デバイスドライバ自身が保管する事を特徴とする。

【０００７】

【実施例】以下、添付図面を参照して本発明に係る好適な一実施例を詳細に説明する。通常、コンピュータとプリンタとの接続は、例えばセントロニクスを代表とする

ケーブルによって物理的に行うが、ローカルエリアネットワーク（ＬＡＮ）に接続されたネットワークプリンタを考えても分かるように、物理的にも論理的にもケーブルが接続点であると考えるのは誤りである。ケーブルは物理的接続点であるが、もし、ケーブルが論理接続点でもあるなら、アプリケーションは個々のプリンタやスキャナの物理的仕様の違いまでも切り替えなければならない事になってしまう。

【０００８】この問題を避けるため、入出力デバイスメーカーにより供給される入出力デバイスをコンピュータに接続する時に、コンピュータのオペレーティングシステム（ＯＳ）にデバイスドライバをインストールしている。実施例では、プリンタやスキャナ等の入出力デバイスとのインターフェースを以下の４項目について仮想化することにより、実現するものである。

（１）論理接続点
（２）標準色空間
（３）仮想色補正（仮想色空間変換）
（４）仮想色空間変換プロトコル

＜論理接続点＞図１は、本発明のシステムブロック図である。同図において、１はスキャナ、２は接続ケーブル、３はスキャナのデバイスドライバ、５は例えばカラーＤＴＰソフトの様な画像処理アプリケーション、６はプリンタのデバイスドライバ、７は接続ケーブル、８はプリンタである。デバイスドライバ３，６はコンピュータ４の中に存在し、互いにソフト的な結合パス９及び１０で接続されている。

【０００９】従来、カラー画像データのインターフェースは入出力デバイス１，８との物理的な接続点（２，７）で規定されているが、実施例では、論理的な接続点（９，１０）で規定する。つまり、デバイスドライバ３は物理的にはコンピュータ４の中に存在するが、論理的にはスキャナ１の一部と考えられ、また、プリンタドライバ６は物理的にはコンピュータ４の中に存在するが、論理的にはプリンタ８の一部と考えられる。

【００１０】従って、コンピュータ４とスキャナ１又はプリンタ８との色空間特性の違いについては、デバイスドライバ３，６とのインターフェース９，１０を論理接続点として、そこで色補正をするとやりやすい。以下これらの点を論理接続点９及び１０と呼ぶ事にする。実施例では、これら論理接続点の左右に標準色空間を設定する。例えば、コンピュータ４がＲＧＢの色空間を用い、プリンタ８がＹＭＣの色空間を用いている場合には、デバイスドライバ６がＲＧＢからＹＭＣへの変換を行う。

【００１１】＜論理接続点における色補正＞この方式は、ＲＧＢやＹＭＣの様に、明確に異なる色空間同士の間の変換だけに適用するものではなく、例えばＲＧＢとＲＧＢ同士の間にも適用できることは言うまでもない。例えば、スキャナ１は通常ＲＧＢ信号を出力することが多く、コンピュータ４もＲＧＢを標準にすることが多い。しかし、両者の白バランスは微妙に違っている事が多く、従来、最終的にはユーザが手動で微調整をしていたか、何もせずに諦めていた。しかるに、スキャナ１の色空間が仮にＲ′Ｇ′Ｂ′だったとすると、コンピュータ４の色空間ＲＧＢに合わせるための補正をどの部分がやらなければならないかと言うと、やはりデバイスドライバ３が行うのが理想的である。つまり、論理接続点において色補正を実施するという形態が最も望ましい。

【００１２】＜標準色空間の導入＞上述の如く、スキャナ１の色空間をＲ′Ｇ′Ｂ′とし、コンピュータ４の色空間をＲＧＢとしたのは適切でない。なぜなら、良く設計されたスタンドアロンのスキャナは、通常その内部で色補正をして出力するから、自分自身は標準のＲＧＢである、と信じて出力しているので、もしかするとコンピュータ４の色空間の方が狂っているかも知れないのである。通常、コンピュータは正しい色空間に則っていると錯覚するのは、コンピュータの内部で扱われる者はデジタルの数値であり、スキャナは光源、撮像素子を含むアナログ装置であるからどうしても分が悪い。ところが、コンピュータ上である値をセットすると、それが何色になるかという事は、実際には分からないのである。なぜなら、コンピュータのデータはＣＲＴディスプレイ上に表示されて初めて人間の目に触れる訳だから、ＣＲＴディスプレイが正確に校正されたものでない限りコンピュータの数値が正しいとは言えないのである。通常、諸々の色補正を行う時に、コンピュータではなく各デバイスを調整するのは、コンピュータが全てのデバイスを制御しているため、コンピュータで色補正を行う訳には行かず、仮にコンピュータは正しいと仮定して使っているに過ぎないのである。

【００１３】＜色補正をコンピュータで行う矛盾＞実際のシステムの構成として、最終的な色補正をコンピュータで行うのは便利だから、しばしばそうした構成になりがちである。しかし、これは上述した論理に照らし合わせるとはなはだ問題である。具体的な例で説明すると、あるアプリケーションでは、コマンドのメニューの中からスキャナを駆動できる様になっている。つまり、アプリケーションが直接スキャナを駆動できるのだからさぞかし便利なはずであるが、実際にはメニューの中におびただしいスキャナ製品のリストが表示され、その中から自分のスキャナを選択してから原稿のスキャンとなる。このスキャナの選択は一回やっておけば良いので、何等問題はなさそうにも思えるが問題も生じる。

【００１４】このように、アプリケーションから諸々のスキャナが選択できるという事は、裏を返せば各種スキャナの諸々のセッティングもアプリケーションの責任でやらなければならないと言う事を意味する。色補正もしかりである。世の中に、白黒２値のスキャナが数種類しかなかった頃ならこれでも良いが、白黒の中間調あり、カラーありという時代になって来るとこのやり方では不

充分である．やはり、デバイスドライバの所で責任の分界点を明確にし、スキャナの色補正はスキャナ側の責任で行わなければならない．即ち、ユーザに取ってたった１台のスキャナであっても、システムとしてはどんなスキャナを持って来ても無調整でつながるような構成になっていなければならない．

【００１５】＜標準色空間の設定＞上述の問題点や矛盾点を解決する為に、実施例では論理接続点において標準色空間を設定している．式１は、色［R′ G′ B′］を色［ＲＧＢ］に変換するための式である．［f_{11}, f_{12}, …, f_{33}］は変換係数マトリックスである．また式２は、色［ＲＧＢ］を色［ｒｇｂ］に変換するための式である．［g_{11}, g_{12}, …, g_{33}］は変換係数マトリクスである．

【００１６】式１の［ＲＧＢ］を式２に代入すると、式３が得られる．ここで、式３の変換係数マトリクスを式４の様に置き換えると、式３は式５の様に表される．即ち、変換係数マトリクス同士の計算を１回やっておけば、２回やるべき色変換のための計算を１回で済ますことができる事が解る．変換係数マトリクス［f_{11}, f_{12}, …, f_{33}］と［g_{11}, g_{12}, …, g_{33}］は簡単のため、それぞれ式６，式７の様に［F］，［G］と表現する．

【００１７】

【数１】

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{pmatrix} \begin{pmatrix} R' \\ G' \\ B' \end{pmatrix} \qquad \text{式１}$$

【００１８】

【数２】

$$\begin{pmatrix} r \\ g \\ b \end{pmatrix} = \begin{pmatrix} g_{11} & g_{12} & g_{13} \\ g_{21} & g_{22} & g_{23} \\ g_{31} & g_{32} & g_{33} \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \qquad \text{式２}$$

【００１９】

【数３】

$$\begin{pmatrix} r \\ g \\ b \end{pmatrix} = \begin{pmatrix} g_{11} & g_{12} & g_{13} \\ g_{21} & g_{22} & g_{23} \\ g_{31} & g_{32} & g_{33} \end{pmatrix} \begin{pmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{pmatrix} \begin{pmatrix} R' \\ G' \\ B' \end{pmatrix} \qquad \text{式３}$$

【００２０】

【数４】

$$\begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} = \begin{pmatrix} g_{11} & g_{12} & g_{13} \\ g_{21} & g_{22} & g_{23} \\ g_{31} & g_{32} & g_{33} \end{pmatrix} \begin{pmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{pmatrix} \qquad \text{式４}$$

【００２１】

【数５】

$$\begin{pmatrix} r \\ g \\ b \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} R' \\ G' \\ B' \end{pmatrix} \qquad \text{式５}$$

【００２２】

【数６】

$$(F) = \begin{pmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{pmatrix} \quad \text{式6}$$

【0023】　　【数7】

$$(G) = \begin{pmatrix} g_{11} & g_{12} & g_{13} \\ g_{21} & g_{22} & g_{23} \\ g_{31} & g_{32} & g_{33} \end{pmatrix} \quad \text{式7}$$

【0024】　　【数8】

$$\begin{pmatrix} r \\ g \\ b \end{pmatrix} = (D)(G)(F)\begin{pmatrix} R' \\ G' \\ B' \end{pmatrix} \quad \text{式8}$$

【0025】図2は、コンピュータ4内の色空間特性を示す模式図であり、図1と同じ構成要素には同じ番号を付与してある。同図において、11はスキャナ固有の色空間特性、12は標準空間特性、13はコンピュータ固有の色空間特性であり、便宜上それぞれ［R′ G′ B′］、［NTSC RGB］、［r′ g′ b′］とする。また、［G］は［R′G′B′］を［NTSC RGB］に変換する為の変換手段（マトリクス）、［F］は［NTSC RGB］を［r′ g′ b′］に変換する為の変換手段（マトリクス）である。

【0026】上述の標準色空間は色彩工学上妥当なものであれば何であってもよいが、画像データの送り側（スキャナ3）も受けて側（コンピュータ5）も共に［RGB］の場合には、［NTSC RGB］が望ましい。送り側では、変換手段［G］によってデバイス固有の色空間特性［R′ G′ B′］を、標準色空間［NTSCRGB］に変換する。そして、受け側では、変換手段［F］によって標準色空間［NTSC RGB］からデバイス固有の色空間［r′ g′ b′］に変換する。このように、標準色空間への／から変換義務は、スキャナ1もコンピュータ4も対等に負っている。

【0027】＜仮想色補正（仮想色空間変換）＞前項の目的を実行する為には、［G］と［F］という2回の変換手段が必要になる。しかし、これは以下の様な問題がある。

（1）システム全体の価格がアップしてしまう。

（2）スループットが悪くなる。

【0028】（3）変換する度に、演算誤差が累積する。

（4）変換する度に、色表現のダイナミックレンジが狭くなる。

このような問題を避けるため、実施例では、仮想色補正（仮想色空間変換）の概念を導入している。仮想色補正／仮想色空間変換（以下「仮想変換」）とは、目的とする標準色空間への変換式と、元のデータをセットで送るだけで、実際には何も演算を施さず、最終的な演算はデータを受け取った側に委ねてしまう方法である。変換手段［G］［F］同士の計算をやってから、それに元の画像データを掛算すれば、計算の手間が1回で済み、しかも丸め等による演算誤差の累積もない。但し、データを受け取った側に演算を委ねられるのは、受け側にその様な機能がある場合のみであり、その様な機能がなければ、送り側で変換して出さなければならない。

【0029】＜仮想色変換空間プロトコル＞図3は、実施例における仮想色空間変換プロトコルの概要を示すものである。送り側と受け側では、デバイスドライバを介して通信を行うので、プロトコルはパケットのやり取りにより実現される。例えば、パーソナルコンピュータの場合にはパケットのポインタのみを渡し、実際にパケットを伝送しなくても良い。

【0030】図4～図7は、伝送される各パケットの構造を示す図である。尚、その構造は同様であり、ここでは図4に示す色空間要求（Color_Space_Request)パケットを例に説明する。各要素は1バイトを基本とし、先頭からパケットID（Packet_ID)41、コマンドID（Command_ID）42、長さ（Length）43、内容（Content)44

の順番にデータが構築される．パケットID４１は、そのパケットの種類を表すコードが入る．コマンドID４２は、以下に続く内容の意味を表すコードが入る．長さ４３は、内容４４のバイト単位の長さを示す．コマンドID４２、長さ４３及び内容４４は、必要に応じて繰り返しても良い．パケットの最後には、終了コード（Terminator）４５が入る．終了コード４５はコマンドIDの一種で、［０］を書く．そして、内容４４は可変長で、その長さは、長さ４３がバイト単位に規定する．尚、内容を可変長にしたのは、拡張性を持たせるためで、固定長であっても良い．

【００３１】ここで、図３に示す仮想色空間変換プロトコルについて説明する．まず送り側に対して色空間要求パケットを送り、自分が送りたい画像の色空間を指定すると共に、相手の色空間が何であるかを問い合わせる．同時に、色変換機能の有無も知らせる．この時、送り側は、例えば図４の例に示す様に、［ＲＧＢ］の色空間に則って送信し、且つ自分には色変換機能が無い［Ｎｏ］事を知らせる．一方、受け側は、要求パケットに対して色空間リスト（Color_Space_List）パケットを返し、自分の受けられる色空間のリストを知らせる．この時、受け側は、例えば図５の例に示す様に、［ＲＧＢ］の色空間に則って受信し、且つ、そこに色変換機能がある［Ｙｅｓ］事を知らせる．

【００３２】これに対し、送り側は、改めて図６に示す色空間選択（Color_Space_Select）パケットを送り、使用する色空間とデータ形式を確定する．例えば、［ＲＧＢ］の色空間データを仮想色補正パラメータ付で送る事を確定する．このパケットには、色空間補正用のフィルタ係数（Filter_Coefficients)が含まれている．受け側は、図７に示す色空間選択確認（Color_Space_Select_Ack）パケットによって使用する色空間とデータ形式を確認する．色空間選択確認パケットはオプションであっても良く、必ずしも確認はしなくても伝送は可能である．

【００３３】次に、スキャナやプリンタがネットワークを介してコンピュータに接続されている場合を説明する．送り側に、例えばアップルコンピュータ（以下アップル）社のマッキントッシュ（以下「SP Client 」と略す）、受け側にスタンドアロンのネットワーク・スキャナプリンタサーバ（以下「SP Server 」と略す）を用いてシステムを構築する場合、通信プロトコルを介して両者を接続することになる．マッキントッシュでは、ローカルトークを用いるのが一般的だが、例えばＳＵＮを代表とするＵＮＩＸ環境と共存させるためには、TCP/IPプロトコルの方が便利である．そのため、マッキントッシュ用のTCP/IPプロトコルであるMacTCPを用いて本実施例を実現する方法を述べる．

【００３４】ここで述べる機能要素は、PrintTCP，SP Client ，SP Server ，MacTCPの４つである．マッキントッシュにインストールするのは、PrintTCPとSP Client及びアップルのMacTCPである．PrintTCPは、マッキントッシュのアプリケーションからTCP/IPを介してイーサネット（Etherenet ）上のスキャナプリンタサーバに印刷する為のスキャナプリンタドライバである．PrintTCPの基本機能は以下の通りである．

（１）プリント時にQuickDraw 描画サブルーチンがコールされた時、それと等価なCaPSL (Canon Printing System Language)コードを生成する．ここで、CaPSLはキヤノン（株）が海外市場向けに開発したプリンタ言語である．

（２）BitMapの中間調画像データ（カラー／白黒）は、オプションとしてＪＰＥＧ規格に準拠したＡＤＣＴ方式によるデータ圧縮を掛けたものを、CaPSL コードとして生成する．

（３）生成したCaPSL コードを、S/P Clientドライバを通じてS/P Serverに伝送する．

【００３５】S/P Clientは、S/P Serverに接続されているプリンタにCaPSL コードを伝送するための通信制御プログラムであり、TCP/IP及びEthernetを介して通信を行う．S/P Clientの基本機能は以下の通りである．

（１）TCP/IPを介してS/P ServerとEnd-to-Endのリンクを張る．

（２）PrintTCPから受け取ったCaPSL データを、S/P Serverに送る．

（３）S/P Serverに原稿のスキャンをさせ、それを受信し、アプリケーションに送る．

【００３６】S/P Serverは、ＳＰサーバ上でデーモンとして常に走っており、クライアントからの受信を待っている．S/P Clientの基本機能は以下の通りである．

（１）S/P Clientから受け取ったCaPSL データをCaPSL インタープリタに渡す．

（２）原稿をスキャンするプログラムを起動し、受け取ったデータをS/P Clientに送る．

【００３７】これらのプログラムの関係を図８の太い四角で囲んだ部分に示す．各プログラムユニットのリソースの形式及び機能を次に示す．

（イ）Printing Manager:System に標準でインストールされている次のプリンタドライバをコールする．type=DRVR ，iPrDrvRef=-3

（ロ）PrintTCP：このプリンタドライバは、初めはSystemにインストールせず、Chooser Documentの形でコードリソースで提供し、Laser WriterやImage Writerと共に、Chooser DAから選択可能にする．

（ハ）S/P Client：S/P Clientドライバは、INIT－31メカニズムによって電源の立ち上げ時に自動的に自分自身をSystemにインストールする．

（ニ）MacTCP：このドライバは、[Control Panel Document]のリソースも有し、Control DAからIPアドレス等の初期値をセットできる．

（ホ）S/P Server：ＦＳＸサーバ側のプログラムであ

り、例えばＵＮＩＸのデーモンの形で常駐している。

【００３８】図８において、２０は市販のＤＴＰアプリケーション、２１はOSに常駐しているPrintingManager、２２はPrintTCP、２３はSP Client、２４はMacTCP、２５はイーサネットボード、２６はＡＤＣＴボードの制御関数、２７はＡＤＣＴ圧縮ボード、２８はイーサネットケーブル、２９はイーサネットボード、３０はＵＮＩＸに常駐するTCP/IPプロトコル、３１はSP Server、３２はCaPSL インタープリタ、３３はＡＤＣＴボードの制御関数、３４はＡＤＣＴ圧縮ボード、３５はプリンタである。２０から２５までがマッキントッシュ、２９から３４までがＮＷＳＰである。２５と２９はケーブル２８を介して通信機能を提供するハードウェアである。実際の論理的通信路は、MacTCP２４とTCP/IP３０によって確保される。SP Client ２３とSP Server ３１は、TCP/IPの提供する汎用の論理通信路上にEnd-to-Endで構築されるプリンタサーバクライアントプロトコルである。このプロトコルの目的は、アプリケーション２０にネットワーク（この場合はイーサネット）を意識させずに、プリンタ３５があたかもローカルなコンピュータに接続されているかの様に見せる為のドライバである。言い換えると、プリンタ３５はアプリケーション２０から見ると、あたかもPrintingManager ２１の下に直接接続されているかの様に見える。

【００３９】図９は、図８のコンピュータ側について、OS３６との関わりをより詳細に説明した図である。図示する様に、PrintingManager ２１はOS３６の一部を構成している。２２と２３，２３と２４，２２と２６はDeviceManager ３７に媒介されて互いに通信する。PrintTCP ２２は、QuickDraw - CaPSL 変換部２２－１とPrinterDriver 部２２－２とに分かれ、２２－１はQuickDraw の描画ルーチンをCaPSLコードに置き換えている所である。これによってQuickDraw の代わりにCaPSL インタープリタを有するSP SeverがQuickDraw をエミュレートできる。各ドライバをアプリケーションとしてではなく、ドライバとして構成する最大のメリットは、インターフェースがOS３６を介してつながるので、スペックが標準化しやすいことである。また、マルチファインダーとは言っても、事実上シングルタスクのOSの場合、ドライバ化する事によって、その部分をマルチタスク化する事ができる。例えばＵＮＩＸの場合には、文字どおり物理デバイスとのインターフェースのみをドライバ化し、あとはデーモンとして簡単にバックグラウンドで走らせる事ができる。

＜SP Client ドライバの形式＞SP Client ドライバは、INIT-31 メカニズムによって電源の立ち上げ時に自動的に自分自身をシステム内にインストールする。ドライバのメモリ占有サイズが大きくなる時は、大部分のコードはコードリソースの形で持ち、オープンされた時にリソースをシステムヒープにロードする形式が望ましい。その場合、Close時にメモリを開放する。ドライバ名はピリオド［．］で始まる。

【００４０】また、SP Client は［コントロールパネルドキュメント］のリソースも有し、コントロールパネルから、ＩＰアドレス等の各種パラメータを設定できる。そのため少なくとも以下に示す様なリソースを持っている。

ＤＩＴＬ　ＩＤ＝－４０６４
ｍａｃｈ　ＩＤ＝－４０６４
ｎｒｃｔ　ＩＤ＝－４０６４
ＩＣＮ＃　ＩＤ＝－４０６４
ＢＮＤＬ　ＩＤ＝－４０６４
ＦＲＥＦ　ＩＤ＝－４０６４
ｃｄｅｖ　ＩＤ＝－４０６４

＜SP Client ドライバのインターフェース＞SP Client は、次のハイレベルデバイスマネジャールーチンを提供する。

【００４１】DriverOpen
DriverClose
Control
FSRead
FSWite
Status
KillIO

上記標準ドライバインターフェースに無いSP Client のサービスは、Controlルーチンが提供する。Control コールのパラメータブロックの持つcs Code を所定の値に設定する事によって、次の様な各種コマンドが利用できる。

【００４２】SPSetInit
SPListen
SPCapability
Color_Space_Request
Color_Space_List
Color_Space_Select
Color_Space_Select_Ack

＜コマンドの説明＞各種コマンドは、表１に示す通りである。

【００４３】
【表１】

DriverOpen	<項目>	<define>	<内容>	<例、コメント>
	パラメータ	host	ホスト名	
		wind	ウィンドウサイズ	(1,2,3,4)
		*buffer	バッファポインタ	(SPOpenの結果が書き込まれる)
	戻り値：	OsErr	結果	
		refnum	ポート参照番号	(以後この番号でやり取りする)
DriverClose	<項目>	<define>	<内容>	<例、コメント>
	パラメータ	refnum	ポート参照番号	
	戻り値：	OsErr	結果	
FSRead	<項目>	<define>	<内容>	<例、コメント>
	パラメータ	refnum	ポート参照番号	
		*buffer	バッファポインタ	(受信データバッファ)
	戻り値：	OsErr	結果	
		RxSize	データサイズ	(実際受信したデータのサイズ)
		EOF	End of File	(データ終りのフラグ)
FSWrite	<項目>	<define>	<内容>	<例、コメント>
	パラメータ	refnum	ポート参照番号	
		*buffer	バッファポインタ	(送信データバッファ)
		size	データサイズ	(送信データの大きさ)
		EOF	End of File	(データ終りのフラグ)
	戻り値：	OsErr	結果	
Status	<項目>	<define>	<内容>	<例、コメント>
	パラメータ	host	ホスト名	
		*buffer	バッファポインタ	(SPStatusの結果が書き込まれる
	戻り値：	OsErr	結果	
Control	<項目>	<define>	<内容>	<例、コメント>
	パラメータ	refnum	ポート参照番号	
		*buffer	バッファポインタ	(データバッファ)
		size	データサイズ	(データの大きさ)
	戻り値：	OsErr	結果	
SPInit	csCode=cSPInit	サーバが最初にこれを発行して、受信の準備をする		
	<項目>	<define>	<内容>	<例、コメント>
	パラメータ	host	ホスト名	
		Wind	ウインドサイズ	(1,2,4,8)
		*buffer	バッファポインタ	(SPOpenの結果が書き込まれる)
	戻り値：	OsErr	結果	
		refnum	ポート参照番号	(以後この番号でやり取りする)
SPListen	csCode=cSPListen	サーバがこれを発行して、受信の確認をする		
	<項目>	<define>	<内容>	<例、コメント>
	パラメータ	refnum	ポート参照番号	
	戻り値：	OsErr	結果	
SPSetStatus	csCode=cSPStatus	サーバがこれを発行して、S/P Serverの設定を変更する		
	<項目>	<define>	<内容>	<例、コメント>
	パラメータ	refnum	ポート参照番号	
		newStatus	新しいステータスの設定値	
	戻り値：	OsErr	結果	

【００４４】<パケットの形状>ＳＰサーバ／クライアントプロトコルの一般的なパケットの形状は、図１０に示す通りであり、図において左側の数字はバイト数である。

<各パケットの機能>そして、ＳＣＣＬパケットとＤＶＣＬパケットの機能は、表２乃至表４に示す通りである。

【００４５】

【表２】

15　　16

<CSSLパケット>

OpenConn	サーバ／クライアントリンクを張り、その後バーチャル通信回線を確保する サーバ／クライアント通信の機能を相手に知らせる
OpenConnReply	サーバ／クライアントリンクの確認 サーバ／クライアント通信の機能の確認 接続番号(Conn_ID) の確立
CloseConn	サーバ／クライアントリンクの切断
CloseConnReply	サーバ／クライアントリンクの切断確認
Data	データ
Ack	データ送達確認
Nack	データ送達確認、フロー制御
Status	サーバ／クライアントリンクの状態の問い合わせ
StatusReply	Statusに対する返事で、RR,RNR,Status ACK,Abort の何れかである
Abort	通信の中断
Control	サーバ／クライアント制御（PrintTCPへのデータ以外の全てのサービス）

【００４６】　　【表３】

＜ＤＶＣＬパケット＞

Init(com_id=fxInit) device=<device_name> direction=<data_direction> PDL=<language> paper=<paper_size> scape=<scape_type> resolution=<dpi> color=<color_space>	：ＳＰサーバの初期化要求 (eg. Print, Scan, Get, Give) (eg. CaPSL, Postscript, HPGL) (eg. A4, A3) (eg. Landscape, Portrait) (eg. 100, 200, 300, 400) (eg. BK, RGB, RGBX, CMYK, LAB, XYZ, YCrCb)
InitAck(com_id=fxInitAck) result=<result_code>	：ＳＰサーバの初期化応答 (eg. noErr, Error)
Scan(com_id=fxScan) paper=<paper_size> scape=<scape_type> resolution=<dpi> color=<color_component>	：スキャナのスキャン開始要求 (eg. A4, A3) (eg. Landscape, Portrait) (eg. 100, 200, 300, 400) (eg. R, G, B, C, M, Y, K, X)
ScanAck(com_id=fxScanAck) result=<result_code> which_image=<image_id>	：スキャナのスキャン開始応答 (eg. noErr, Error) (eg. 1, 2, 3...)
Print(com_id=fxPrint) paper=<paper_size> scape=<scape_type> resolution=<dpi> color_compo=<color_component> pages=<number_of_pages>	：プリンタのプリント開始要求 (eg. A4, A3) (eg. Landscape, Portrait) (eg. 100, 200, 300, 400) (eg. R, G, B, C, M, Y, K, X) (eg. 1-99)
PrintAck(com_id=fxPrintAck) result=<result_code> which_image=<image_id> queue_number=<number>	：プリンタのプリント開始応答 (eg. noErr, Error) (eg. 1, 2, 3...) (eg. 1, 2, 3...all)
Capability(com_id=fxCapability)	：ＳＰサーバの機能の確認要求
CapabilityAck(com_id=fxCapabilityAck) device=<device_name> direction=<data_direction> PDL=<language> paper=<paper_size> scape=<scape_type> color=<color_space>	：ＳＰサーバの機能の確認応答 (eg. Print, Scan, Get, Give) (eg. CaPSL, Postscript, HPGL) (eg. A4, A3) (eg. Landscape, Portrait) (eg. BK, RGB, RGBX, CMYK, LAB, XYZ, YCrCb)
SetArea(com_id=fxSetArea) area=<print_area>	：原稿の画像有効領域の設定要求 (eg. top, left, bottom, right)
SetAreaAck(com_id=fxSetAreaAck) which_image=<image_id> result=<result_code>	：原稿の画像有効領域の設定応答 (eg. 1, 2, 3...) (eg. noErr, Error)
SetColor(com_id=fxSetColor) which_image=<image_id> color=<color_space> color_compo=<color_component>	：色指定要求 (eg. 1, 2, 3...) (eg. BK, RGB, RGBX, CMYK, LAB, XYZ, YCrCb) (eg. R, G, B, C, M, Y, K, X)
SetColorAck(com_id=fxSetColorAck) result=<result_code>	：色指定応答 (eg. noErr, Error)
BufFlush(com_id=fxBufFlush) which_image=<image_id>	：画像メモリのクリア要求 (eg. 1, 2, 3...)
BufFlushAck(com_id=fxBufFlushAck) result=<result_code>	：画像メモリのクリア応答 (eg. noErr, Error)

【００４７】　　【表４】

ClearQueue(com_id=fxClearQueue) queue_number=<number>	:プリンタキューのクリア要求 (eg.1,2,3...all)
ClearQueueAck(com_id=fxClearQueueAck) result=<result_code>	:プリンタキューのクリア確認 (eg.noErr,Error)
Comp(com_id=fxComp) which_image=<image_id> type=<compression_type>	:画像の圧縮要求 (eg.1,2,3...) (eg.JPEG,MH,MR,MMR)
CompAck(com_id=fxCompAck) which_image=<image_id> result=<result_code>	:画像の圧縮確認 (eg.1,2,3...) (eg.noErr,Error)
DeComp(com_id=fxDeComp) which_image=<image_id> type=<compression_type>	:画像の伸長要求 (eg.1,2,3...) (eg.JPEG,MH,MR,MMR)
DeCompAck(com_id=fxDeCompAck) which_image=<image_id> result=<result_code>	:画像の伸長確認 (eg.1,2,3...) (eg.noErr,Error)
DIR(com_id=fxDIR)	:ディレクトリの要求
DIRAck(com_id=fxDIRAck) directory=<directory>	:ディレクトリの応答 (eg./home/user_name/file_name)
CD(com_id=fxCD) directory=<directory>	:ディレクトリ変更要求 (eg./home/user_name,...)
CDAck(com_id=fxCDAck) result=<result_code>	:ディレクトリ変更応答 (eg.noErr,Error)
Get(com_id=fxGet) data=<data>	:ファイル送信要求
GetAck(com_id=fxGetAck) result=<result_code>	:ファイル送信確認 (eg.noErr,Error)
Put(com_id=fxPut) data=<data>	:ファイル受信要求
PutAck(com_id=fxPutAck) result=<result_code>	:ファイル受信確認 (eg.noErr,Error)
Cancel(com_id=fxCancel) result=<result_code>	:スキャナサーバ操作，セッティングキャンセル (eg.noErr,Error)
Color_Spae_Request(com_id=cColor_Spae_Request) Color_Spae Capability	:送り側の色空間の要求と色処理機能 色空間の名称 色処理能力
Color_Spae_List(com_id=cColor_Spae_List) Color_Spae Capability	:受け側の色空間のリストと色処理機能 色空間の名称 色処理能力
Color_Spae_Select(com_id=cColor_Spae_Select) Color_Spae Color_Method	:送り側の色空間の指定と色処理方法の指定 色空間の名称 色処理方法
Color_Spae_Select_Ack(com_id=cColor_Spae_Select_Ack) Color_Spae Color_Method	:受け側の色空間の確認と色処理方法の確認 色空間の名称 色処理方法

【００４８】図１１～図１２は、これらのパケット及びコマンドのシーケンスを示す図である。図１１はクライアントとサーバ間でやり取りするフローに着目して書いたものであり、図１２はクライアント側で、MacTCPとSP Client の間のフローに着目して書いたものである。アプリケーションからは標準のファイルアクセスと同じ、Open, Read, Write, Close のコマンドでアクセス出来る様に設計している。まずアプリケーションからPrOpen

の関数を呼ぶと、SP Client にSPOpenコマンドが届く．これにより、SP Client はConnect コマンドによってTCPActivateOpen 関数を発行し、TCP/IPを接続する．SP Client は、関数の戻り値がnoErr であればTCP/IPの呼が正常に確立したことを示す．SP Client は確立したＴＣＰリンクの上でSP Client ／SPServer 間のリンクを貼るために、引き続きOpenConnパケットをSP Server に送る．SP Server は、プリンタに問題がなければOpenConn Reply を返してセッションリンクの成立を確認する．その後、サーバ側に固有のパラメータを初期化するためにInitパケットとControl パケットを送り、次に色空間制御のためにColor_Space_Request パケットを発行する．その後のやり取りは、先に説明した通りである．これらのSP Server ／SP Client パケットは、すべてTCP/IPプロトコルのTCPSend またはTCPRv パケットによって伝送される．

【００４９】＜応用例＞図１３及び図１４に本発明の応用例を示す．図１３はＣＲＴモニタに画像情報を表示しているところを示している．同図において、５０はＣＲＴモニタ、５１は管面に正しいＮＴＳＣ　ＲＧＢカラーを表示するための補正された色信号、［Ｄ］は補正係数マトリクス、５２は接続ケーブルである．この例の様に、人間の目に正しくＲＧＢ信号を見せるためには、ガンマ（γ）補正等の色補正を行う事が必要となる．図１３では、この補正をコンピュータ側で行っている例である．この様な補正をＣＰＵで行うのは時間がかかるので、通常はモニタ５０側で行うことになる．その様子を図１４に示す．補正係数マトリクス［Ｄ］はモニタ側で掛けられる．通常、モニタの色補正回路は、ハードウェアで構成するか、ＤＳＰ(Digital Signal Prosessor)の様な特殊な高速なプロセッサで行うので、極めて高速な処理が可能である．何れの場合でも、コンピュータ側から見たＣＲＴは、従来はパッシブな受動デバイスであって、与えたＲＧＢの信号を忠実に再現する努力を原っているだけであった．言い換えると、補正係数マトリクス［Ｄ］は、ＣＲＴモニタ５０に含まれる専用の色処理系であり、これをコンピュータが（設置調整の時は例外として）日常の使用時に頻繁にダイナミックに利用するという概念はなかった．

【００５０】本発明に係るデバイスインディペンデントな色処理方法を用いれば、インターフェースはあくまでも仮想の標準色空間で渡しながら、実際には、式４や図２で説明した様なスキャナ等の入力側の色補正係数をも、画像信号とは別々にプロトコルで送る事により、コンピュータの本体のＣＰＵでは画像処理のための演算を一度もやらずにＣＲＴモニタに渡すことされ可能となる．その様子を式８に示す．［Ｆ］はスキャナ固有の色特性を標準に合わせる補正係数、［Ｇ］はコンピュータ内部での補正係数、［Ｄ］はモニタ固有の色特性を標準に合わせるための補正係数である．ＣＲＴモニタが本発明の様にアクティブなデバイスならば、式８の様に色補正演算を最終デバイス内で一気に演算させることさえ出来る．

【００５１】これによって、デバイスインディペンデントな概念を守りつつ、且つハードウェア資源を有効に利用し、しかも高速で演算の累積誤差の少ない色処理系が実現できる．このように、フィルタ演算をカスケードにつなげて行って、最後に一気に演算するというアルゴリズムが成り立つのは、［ＲＧＢ］，［ＸＹＺ］，［ＣＭＹ］，［ＹＩＱ］の様に、三原色（混色系）の色空間表現の相互間の時に容易であり、［Ｌ* ａ* ｂ* ］の様に、ＸＹＺから計算によって求められた色空間表現（顕色系）との間の変換では、やや困難であり、一旦その色空間に変換した方が速い場合もある．

【００５２】＜色校正＞コンピュータ上で処理したカラー画像を印刷する場合、ＣＲＴモニタ上の画像の色と、印刷結果が正しく一致しているかどうかと言うことは、印刷の専門家にとっては深刻な問題である．このため、コンピュータでカラー画像を扱うシステムでは、次の方法により、モニタとプリンタを校正している．

【００５３】モニタを校正する標準的な方法は、ＲＧＢの各色成分が"０"から"１００"まで変化するとして、まず［Ｒ，Ｇ，Ｂ］＝［１００，０，０］の信号を入れて赤を表示し、それを分光光度計で測定する．測定結果は［ＸＹＺ］で得られるので、これを計算で［ＲＧＢ］に変換し、［１００，０，０］になるように［Ｒ］信号の利得を調整する．同様に［Ｒ，Ｇ，Ｂ］＝［１００，１００，１００］の信号を入力し、分光光度計の出力が［１００，１００，１００］になる様に微調整をする．次に、［Ｒ，Ｇ，Ｂ］＝［７０，３０，３０］のごとき中間色を入力して調整し、最後に白バランスとして［Ｒ，Ｇ，Ｂ］＝［７０，７０，７０］を入力して調整を完了する．プリンタについても、基本は同じである．もちろん、これ以外にも色々なやり方がある．

【００５４】以上でモニタとプリンタが、それぞれ単独に調整できたことになる．以後、この調整値はいじらない．ところが、これでもＣＲＴモニタとプリント結果が異なることが普通である．印刷の専門家が深刻な問題であると感じるのは、ここからである．この問題を解決するための便法として、電子色見本が導入された．電子色見本とは、印刷インキメーカーが、自社の代表的インキの印刷見本と同じ色がコンピュータ上でも得られる様に、コンピュータ用の色見本（カラー画像データ）とセットで供給しているものである．コンピュータ上でカラー原稿を作成し、最終的にそれを印刷しなければならない時は、結局は電子色見本を参照しなければ正確な仕事はできない．電子色見本は、アプリケーションの形で提供されるか、市販の有名アプリケーションのドキュメントファイルとして本表とを一致させるために、アプリケーションが持つ色相微調整機能を調整したくなる．しか

し、アプリケーションでこれを始めてしまうと、複数のアプリケーションと複数のインキメーカー、複数の色見本との組み合わせを考えると、すぐに収拾がつかなくなる事が分かる。

【００５５】調整値の設定は、専用のユーティリティプログラムで行っても、ＤＴＰアプリケーションを使って行っても良い。ただし、調整結果はアプリケーションが管理すべきではなく、デバイスドライバに引き渡してそれに管理させるべきである。さもなければ、アプリケーション毎、色見本毎に色調整を行わなければならなくなる。実施例によれば、デバイスドライバに引き渡すカラー画像信号は、仮想の標準カラーを渡す事ができるので、アプリケーション側で色合わせを行う必要はなく、デバイスドライバ自身に行わせることができる。またデバイスドライバは色合わせの情報を、印刷メーカーのインキ毎にファイルにセーブして管理する。日本で入手可能な電子色見本は３種類だから、これだけ切り替えられれば良い。そうすることによって、どんなアプリケーションからでも同じ環境で色見本を参照可能になる。マッキントッシュの場合、デバイスドライバがｃｄｅｖリソースを持つことができる。このメカニズムを用いれば、アプリケーションを起動したままコントロールパネルから色補正をかけることができ、しかも、いったん設定した設定値は全てのアプリケーションに共通となる。

【００５６】上述したデバイスインディペンデントな色処理方法を用いれば、アプリケーションとデバイスドライバが通信できるので、コントロールパネルを開くという面倒な操作をしなくとも、アプリケーションから直接デバイスドライバの設定値を制御できる。図１５にその関係を示す。同図において、３はコンピュータ、６０はプリンタ、６１はプリンタ６０を制御する為のデバイスドライバでコンピュータ３に常駐する。６２は特定の電子色見本に対する校正データのファイル、６３は第２の電子色見本に対する校正データのファイルである。各校正データのファイルは、直接的にはデバイスドライバ６１が管理、使用するが、間接的には図の様にアプリケーションが制御するものであっても良く、また例えばｃｄｅｖから制御しても良い。

【００５７】以上述べたように、実施例におけるデバイスインディペンデントな色処理方法を用いれば、概念上は仮想の標準色空間で渡すので、入力デバイスから受け取った画像データを、常に標準色空間を基準に処理する事ができ、実際にはそのような演算は行うことなく出力デバイスに渡す事もできる。従って、演算の処理速度は向上し、演算による丸め誤差の累積もなく、デバイスインディペンデントな概念を守りつつ、且つハードウェア資源を有効に利用できる色処理系が実現できる。

【００５８】尚、本発明は、複数の機器から構成されるシステムに適用しても、１つの機器から成る装置に適用しても良い。また、本発明はシステム或いは装置にプログラムを供給することによって達成される場合にも適用できることは言うまでもない。

【００５９】

【発明の効果】以上説明したように、本発明によれば、異なる色空間を有する複数のデバイスを１つの仮想デバイスとして利用する事ができる。

【図面の簡単な説明】

【図１】実施例でのシステム構成を示す概略ブロック図である。

【図２】実施例でのデバイスドライバ部分を説明するための図である。

【図３】実施例でのデバイスインディペンデントな色空間制御プロトコルである。

【図４】実施例での色空間制御プロトコルに用いるパケットの構造図である。

【図５】実施例での色空間制御プロトコルに用いるパケットの構造図である。

【図６】実施例での色空間制御プロトコルに用いるパケットの構造図である。

【図７】実施例での色空間制御プロトコルに用いるパケットの構造図である。

【図８】サーバ／クライアントシステムに応用した例である。

【図９】クライアント側の詳細な構成図である。

【図１０】色空間制御プロトコルに用いる一般的なパケットの構造図である。

【図１１】サーバ／クライアント間のシーケンスの例である。

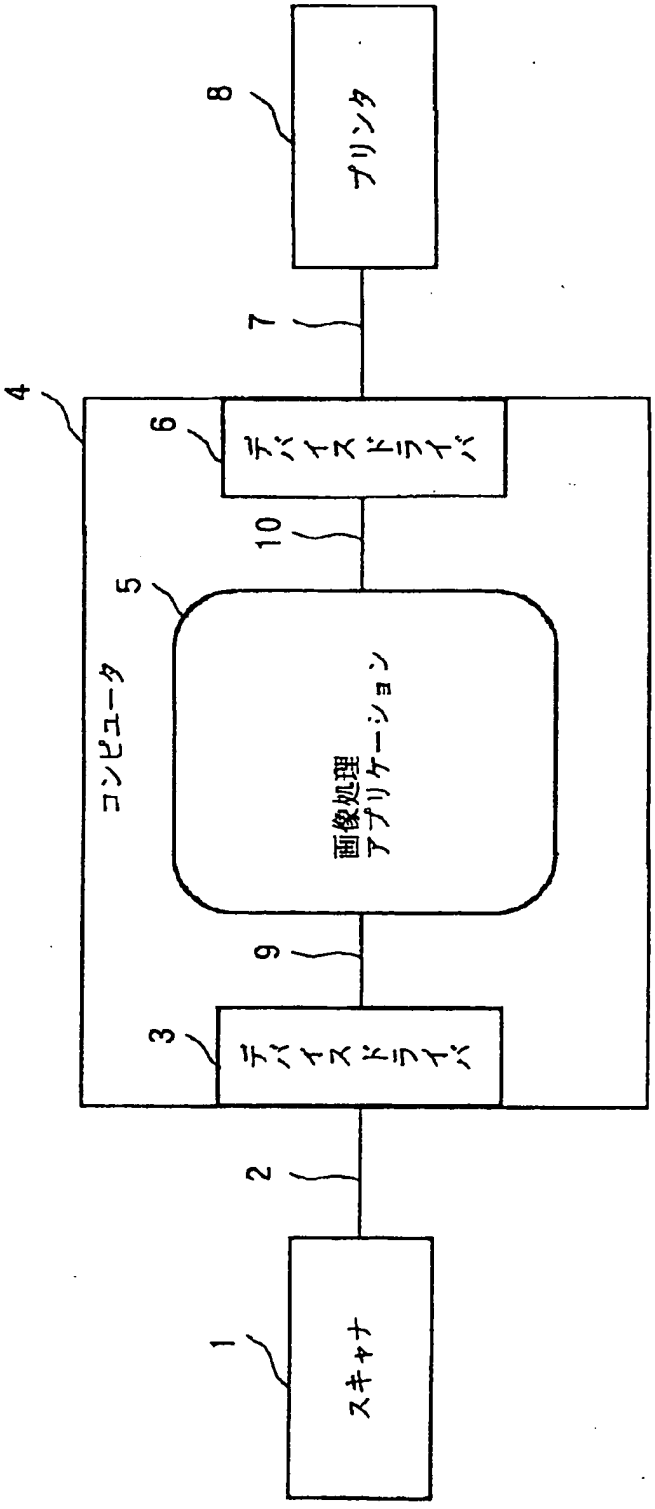
【図１２】クライアント側のシーケンスの詳細な例である。

【図１３】モニタにおける色補正の例である。

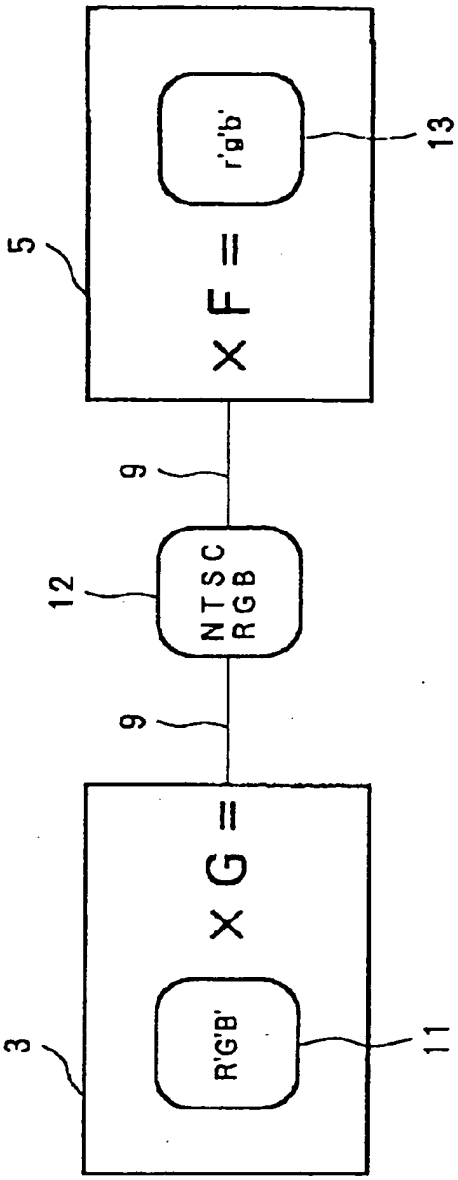
【図１４】モニタにおける色補正の例である。

【図１５】色校正データの管理方法を示す図面である。

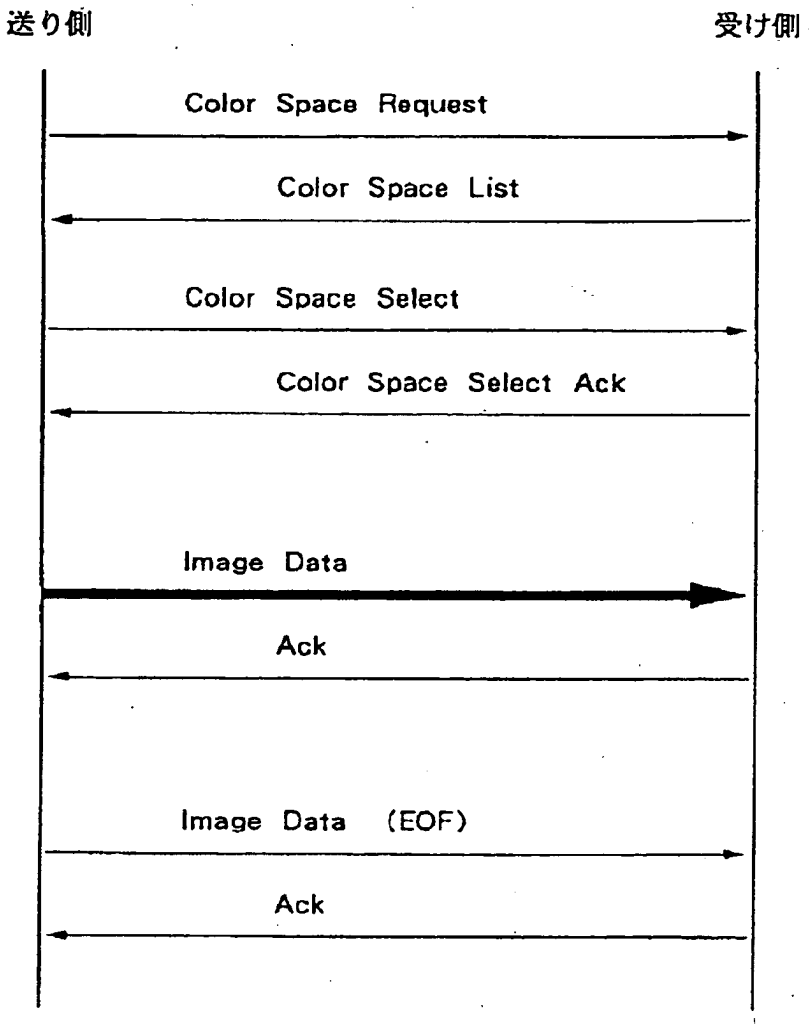
【図1】


スキャナ
1
2
デバイスドライバ
3
コンピュータ
4
画像処理
アプリケーション
5
デバイスドライバ
6
7
プリンタ
8
9
10

【図2】


R'G'B'
XG=
r'g'b'
XF=
NTSC
RGB
3
5
9
11
12
13

【図3】


送り側
受け側
Color Space Request
Color Space List
Color Space Select
Color Space Select Ack
Image Data
Ack
Image Data (EOF)
Ack

【図4】

Color Space Request パケット

Packet ID	Color Space Request	41
Command ID	Color Space	42
Length	3	43
Content (Color Space)	"RGB"	44
Command ID	Capability	42'
Length	1	43'
Content (Capability)	No	44'
Terminator	0	45

【図5】

Color Space List パケット

Packet ID	Color Space List
Command ID	Color Space
Length	3
Content (Color Space)	"RGB"
Command ID	Capability
Length	1
Content (Capability)	Yes
Terminator	0

【図6】

Color Space Select パケット

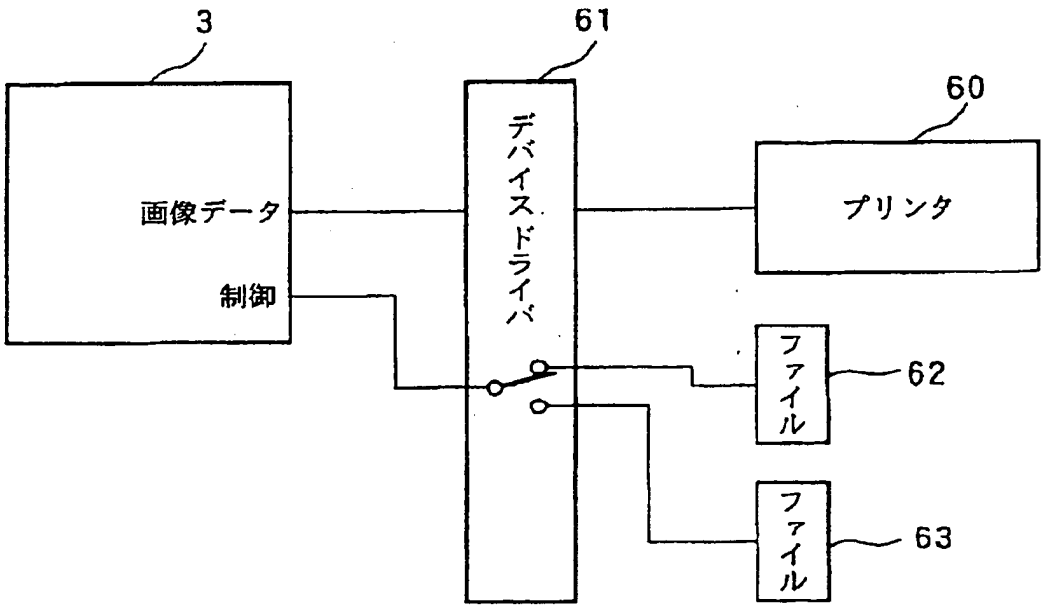
Packet ID	Color Space Select
Command ID	Color Space
Length	3
Content (Color Space)	"RGB"
Command ID	Color Method
Length	1
Content (Color Method)	with Filter
Command ID	Filter
Length	9
Content (Filter)	Filter Coefficients
Terminator	0

【図7】

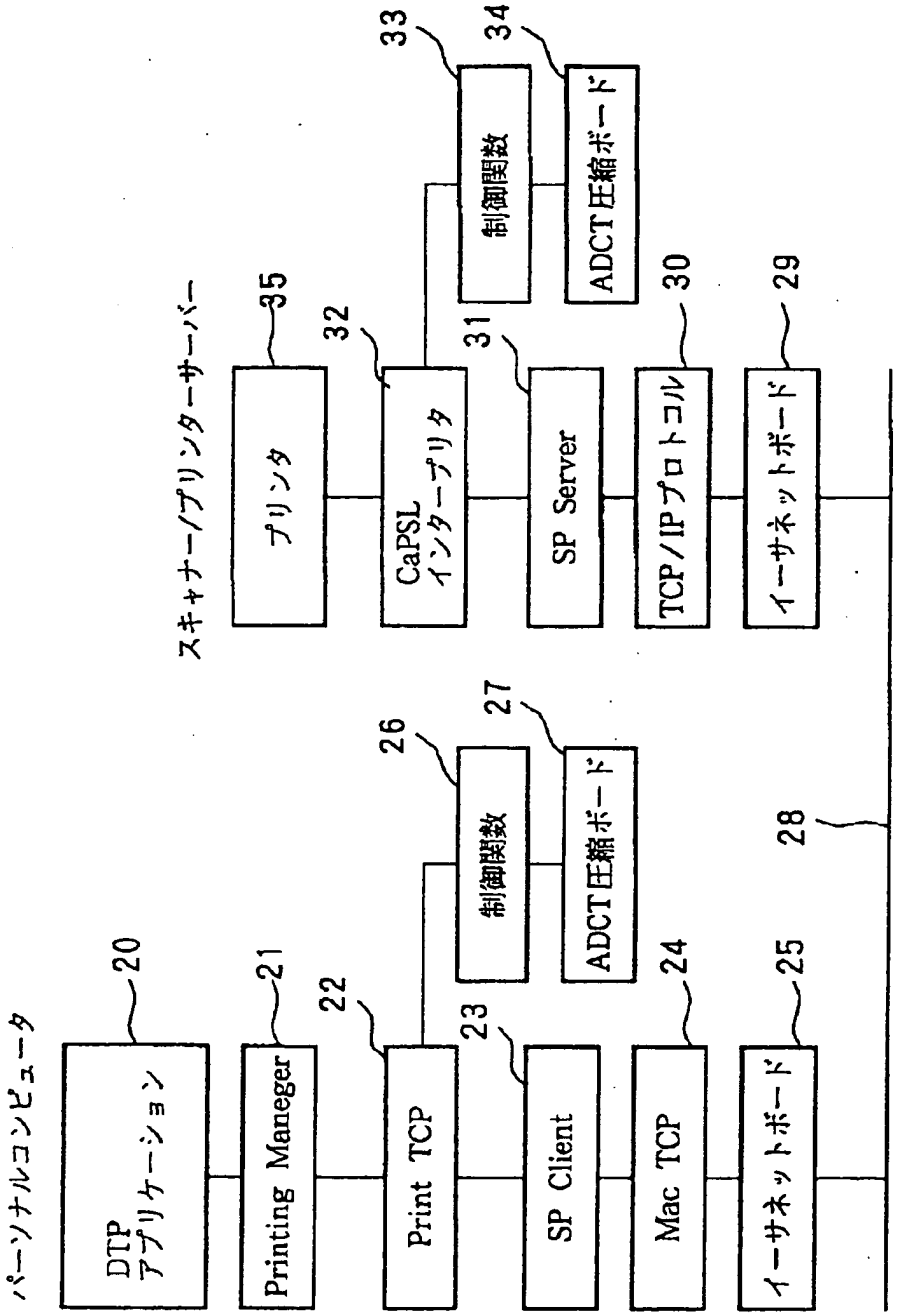
Color Spacc Select Ack パケット

Packet ID	Color Space Select Ack
Command ID	Color Space
Length	3
Content (Color Space)	"RGB"
Command ID	Color Method
Length	1
Content (Color Method)	With Filter
Terminator	0

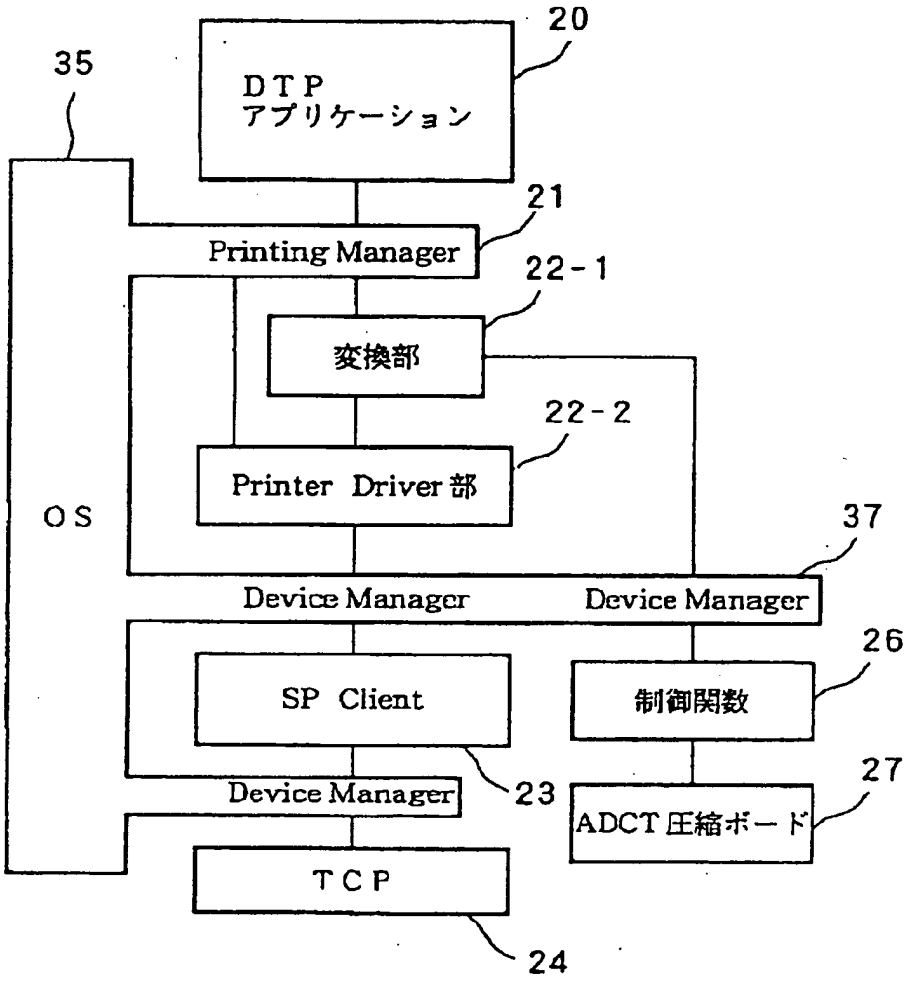
【図15】



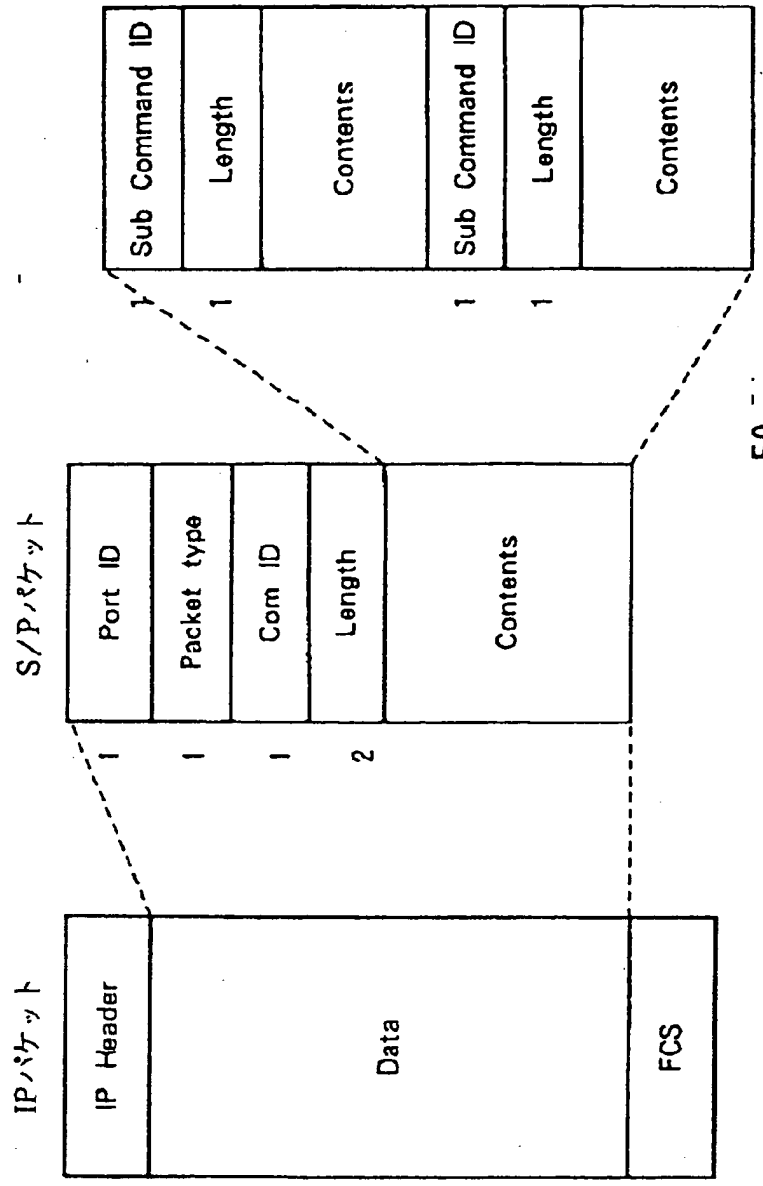
【図8】


パーソナルコンピュータ
DTP アプリケーション
20
Printing Maneger
21
Print TCP
22
SP Client
23
Mac TCP
24
イーサネットボード
25
28
制御関数
26
ADCT圧縮ボード
27
スキャナー/プリンターサーバー
プリンタ
35
CaPSL インタープリタ
32
SP Server
31
TCP/IPプロトコル
30
イーサネットボード
29
制御関数
33
ADCT圧縮ボード
34

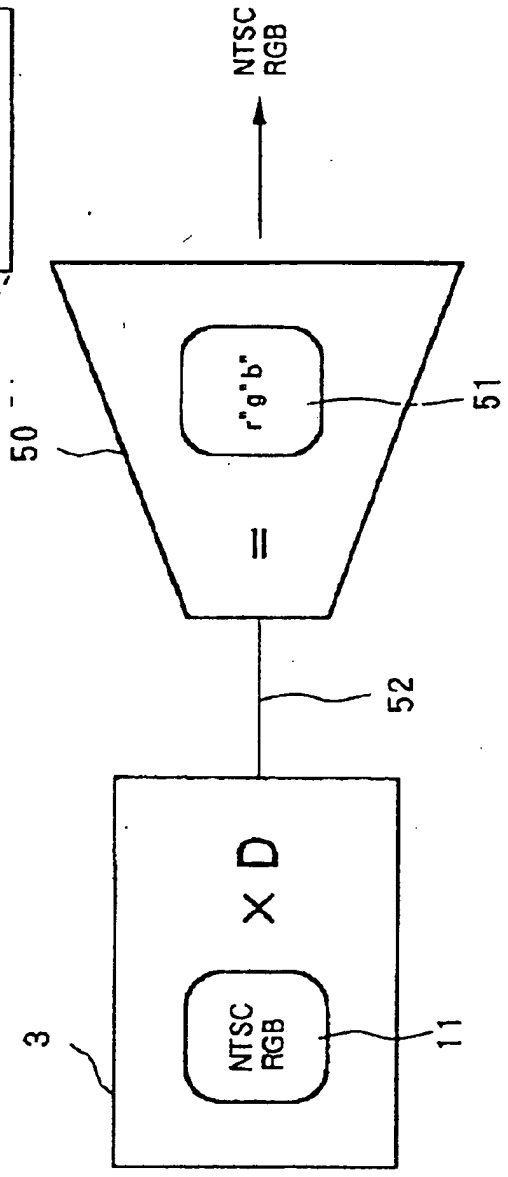
【図９】


20
35
ＤＴＰ
アプリケーション
21
Printing Manager
22-1
変換部
22-2
Printer Driver 部
ＯＳ
37
Device Manager
Device Manager
26
SP Client
制御関数
23
Device Manager
27
ADCT 圧縮ボード
ＴＣＰ
24

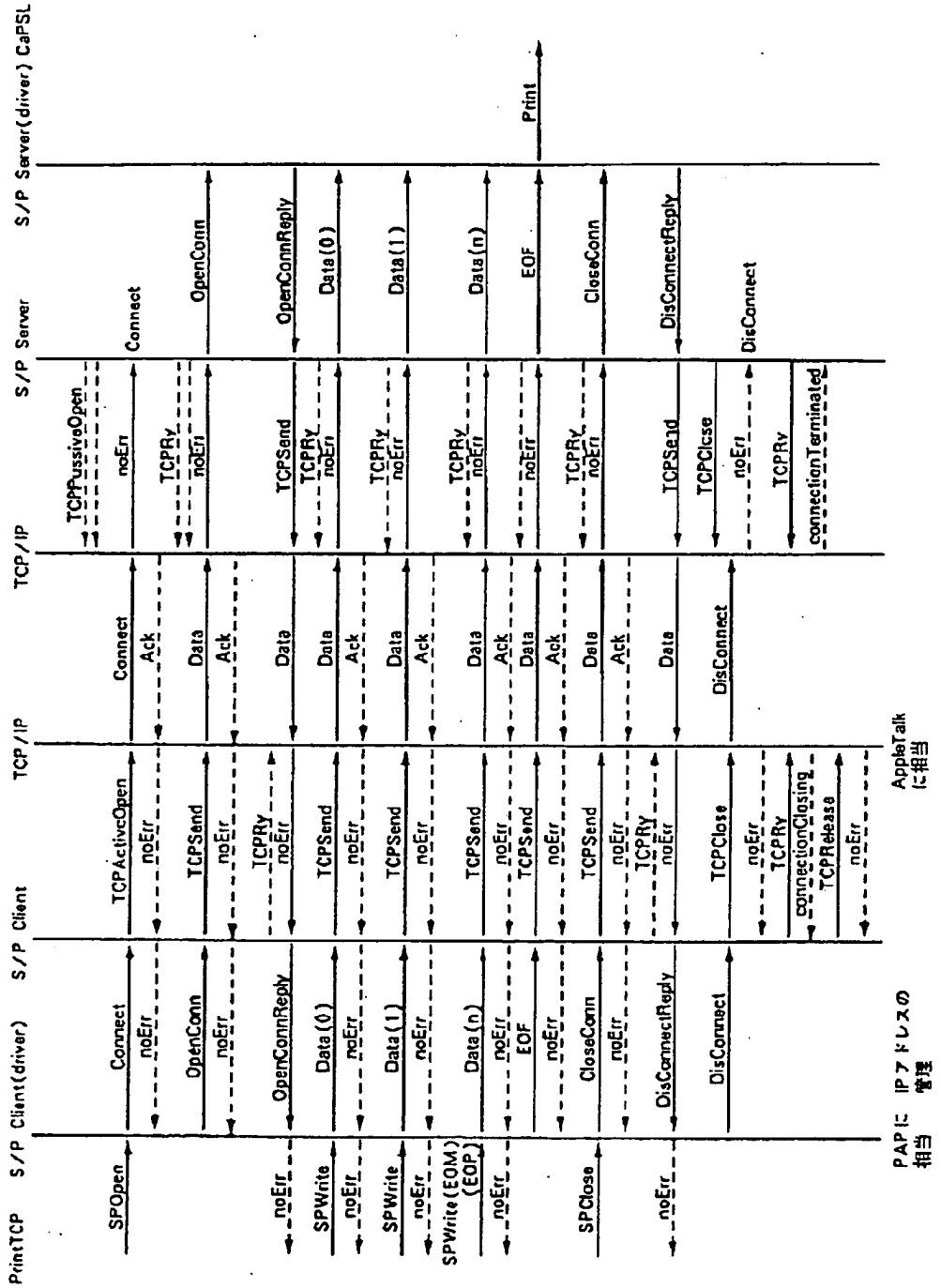
【図10】


IPパケット
IP Header
Data
FCS
S/Pパケット
Port ID
Packet type
Com ID
Length
Contents
1
1
1
2
Sub Command ID
Length
Contents
Sub Command ID
Length
Contents
1
1
1
1

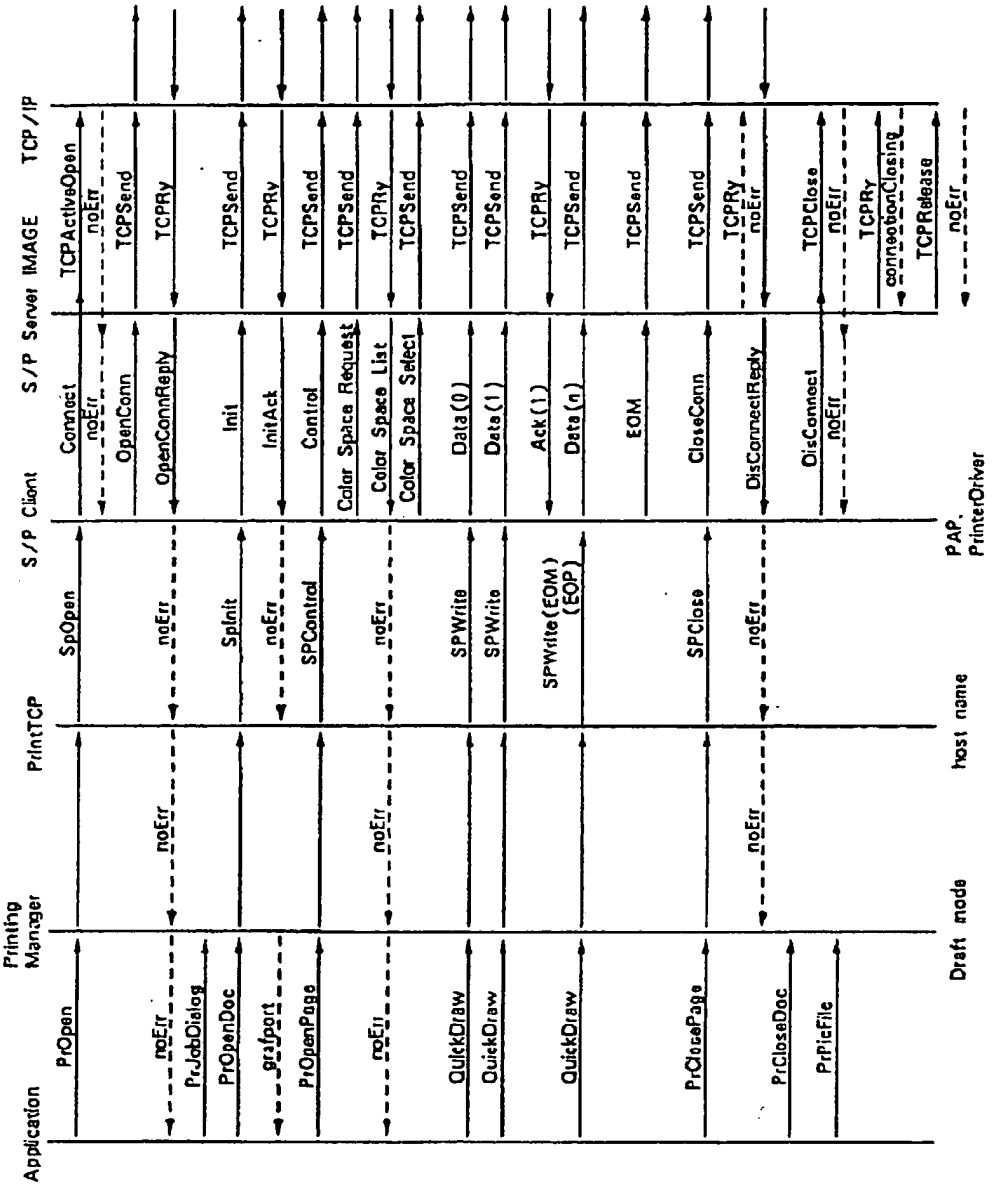
【図13】


NTSC RGB
r"g"b"
51
50
52
× D
NTSC RGB
11
3

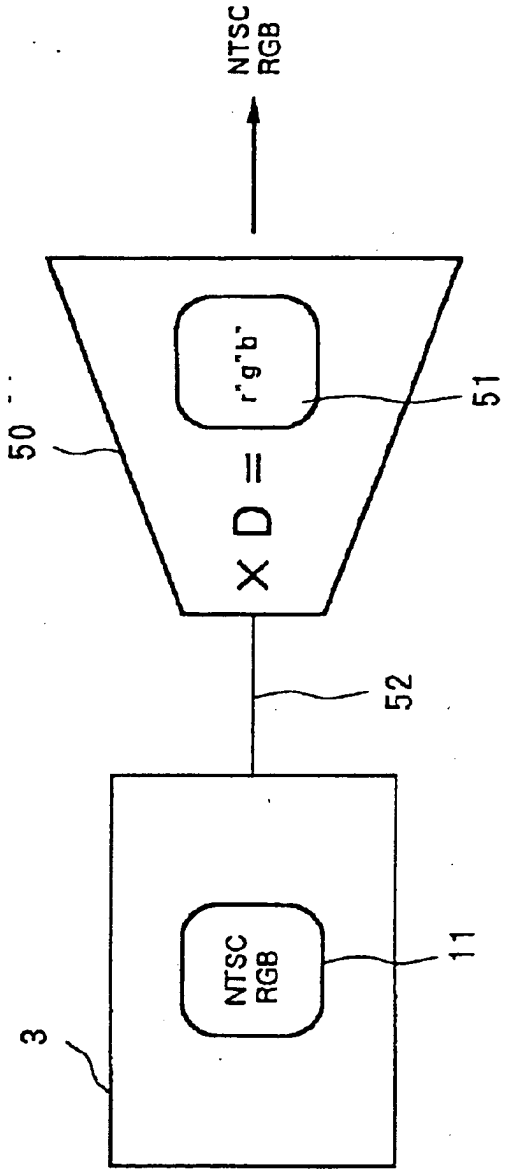
【図11】


PrintTCP
S/P Client(driver)
S/P Client
TCP/IP
TCP/IP
S/P Server
S/P Server(driver) CaPSL
SPOpen
Connect
TCPActiveOpen
noErr
TCPPassiveOpen
noErr
Connect
OpenConn
TCPSend
noErr
TCPRcv
noErr
OpenConn
noErr
TCPRcv
noErr
OpenConnReply
noErr
TCPSend
OpenConnReply
SPWrite
Data(0)
noErr
TCPSend
noErr
TCPRcv
noErr
Data(0)
noErr
TCPRcv
noErr
Data(1)
SPWrite
Data(1)
noErr
TCPSend
noErr
TCPRcv
noErr
Data(1)
noErr
TCPRcv
noErr
Data(n)
SPWrite(EOM)
(EOP)
Data(n)
noErr
TCPSend
noErr
TCPRcv
noErr
Data(n)
noErr
TCPRcv
noErr
EOF
noErr
TCPSend
noErr
TCPRcv
noErr
EOF
noErr
TCPRcv
noErr
Print
SPClose
CloseConn
noErr
TCPSend
noErr
TCPRcv
noErr
CloseConn
noErr
TCPRcv
noErr
DisConnectReply
noErr
TCPSend
DisConnectReply
DisConnect
noErr
TCPClose
noErr
TCPRcv
DisConnect
noErr
TCPClose
noErr
TCPRcv
connectionTerminated
connectionClosing
TCPRelease
noErr
Connect
Ack
Data
Ack
Data
Ack
Data
Ack
Data
Ack
Data
Ack
Data
Ack
Data
DisConnect
AppleTalk
に相当
PAPに IPアドレスの
相当 管理

【図12】


Application
Printing Manager
PrintTCP
S/P Client
S/P Server IMAGE
TCP/IP
Draft mode
host name
PAP PrinterDriver
PrOpen
noErr
PrJobDialog
PrOpenDoc
grafport
PrOpenPage
noErr
QuickDraw
QuickDraw
QuickDraw
PrClosePage
PrCloseDoc
PrPicFile
SpOpen
noErr
SpInit
noErr
SPControl
noErr
SPWrite
SPWrite
SPWrite(EOM)(EOP)
SPClose
noErr
Connect
noErr
OpenConn
OpenConnReply
Init
InitAck
Control
Color Space Request
Color Space List
Color Space Select
Data(0)
Data(1)
Ack(1)
Data(n)
EOM
CloseConn
DisConnectReply
DisConnect
noErr
TCPActiveOpen
noErr
TCPSend
TCPRcv
TCPSend
TCPRcv
TCPSend
TCPSend
TCPRcv
TCPSend
TCPSend
TCPRcv
TCPSend
TCPSend
TCPRcv
noErr
TCPClose
noErr
TCPRcv
connectionClosing
TCPRelease
noErr

【図14】



フロントページの続き

(51) Int. Cl.[6]	識別記号	庁内整理番号	F I	技術表示箇所
9/64		Z 8942-5C		